CLAIMS:

1. A system, comprising:

user directives provided to indicate user desired actions;

instruction information provided to define a suite of instructions; and

a SBE generation tool arranged to generate a software built-in self-test engine (SBE) based on the user directives, the instruction information and device constraints, for subsequent storage on-board of a complex device under test (DUT) and activation of a re-generative functional test on the complex device under test (DUT).

2. The system as claimed in claim 1, wherein said SBE generation tool comprises:

a random instruction test generator (RIT-G) composer arranged to receive the user directives and the instruction information and generate a compact RIT-G code;

a test execution directive composer arranged to receive the user directives and the device constraints and create a run time environment needed to enable the re-generative functional test to repeatedly generate functional tests and execute generated tests on-board the complex device under test (DUT);

a test result compaction module composer arranged to generate a test result compaction module code; and

a code merger arranged to merge code from the RIT-G composer, the test execution

directive composer and the test result compaction module composer to generate the software built-in self-test engine (SBE).

3. The system as claimed in claim 1, wherein said SBE is merged with an expected test result and then loaded on-board a complex device under test (DUT) so as to activate a re-generative functional test on the complex device under test (DUT) and make a comparison between test results of the re-generative functional test and the expected test result to check for design validations and/or manufacturing defects.

4. The system as claimed in claim 3, wherein said expected test result is obtained from computer modeling of the complex device under test (DUT) or from a known good device.

5. The system as claimed in claim 2, wherein said SBE generation tool is a software tool installed on a system for generating the software built-in self-test engine (SBE), and wherein individual components of said SBE generation tool, including the random instruction test generator (RIT-G) composer, the test execution directive composer, the test result compaction module composer, and the code merger, are software modules written in any computer language.

6. The system as claimed in claim 5, wherein said SBE generation tool is provided on a computer readable medium.

7. The system as claimed in claim 2, wherein said SBE generation tool is a hardware implementation installed in the system for generating the software built-in self-test engine (SBE).

8. The system as claimed in claim 2, wherein said run time environment includes a test execution environment which employs an exception handler for handling illegal conditions such as undesirable memory accesses, deadlock, shut-down, and infinite loops, and a RIT environment which provides equivalent operating system (OS) functions needed by the RIT generator to generate the re-generative functional test.

9. The system as claimed in claim 2, wherein said compact RIT-G code produced is a C-language program which is compiled by a C-compiler to produce an assembly language version of the RIT-G code, and when the run time environment, the test result compaction module code and the assembly language version of the RIT-G code are assembled by an assembler, a single program indicating the SBE in the target DUT's object code is obtained.

10. The system as claimed in claim 9, wherein said compact RIT-G code includes an instruction generation module for generating individual instructions during testing application.

11. The system as claimed in claim 1, wherein said software built-in self-test engine

(SBE) as generated comprises:

a RIT generator configured with compact RIT machine code that can reside on-board the complex device under test (DUT) for generating the re-generated functional test;

a test program execution module configured with test execution directives for providing a run time environment to store and run the re-generated functional test; and

a test result compaction module configured with compression machine code that compresses test results of the re-generated functional test for storage on-board the complex device under test (DUT).

12. The system as claimed in claim 11, wherein said test execution environment employs an exception handler for handling illegal conditions such as undesirable memory accesses, deadlock, shut-down, and infinite loops.

13. The system as claimed in claim 1, wherein said complex device under test (DUT) indicates a microprocessor.

14. The system as claimed in claim 13, wherein, when test patterns of the SBE are applied to the microprocessor from an on-board memory, the microprocessor performs the following:

beginning a set-up for executing test patterns;

executing the test patterns to generate a series of test sequences and associated data for respective test sequences;

running the test sequences, and at the end of the test sequences, obtaining the test results for storage in the on-board memory; and

dumping the test results of the test patterns for making a comparison with the expected test result to check for design validations and/or manufacturing defects.

15. The system as claimed in claim 14, wherein said software built-in self-test engine (SBE) is programmed to generate and execute one or more ("N") instruction sequences, each sequence being executed on one or more (M) data sets, where N and M represent an integer no less than "1" and are user-specified numbers used in generating the SBE by the SBE generation tool.

16. The system as claimed in claim 15, wherein said software built-in self-test engine (SBE) is further programmed to generate one or more signatures to provide a unique identification of the test result of each test sequence and indicate whether the test result of a particular test sequence is "good" or "bad".

17. A computer readable medium having stored thereon a software built-in self-test engine (SBE) generation software tool which, when executed by a host system, causes the system

to:

demanding inputs of user directives indicating user desired actions;

obtaining instruction information provided to define a suite of instructions; and

generating a software built-in self-test engine (SBE) based on the user directives, the instruction information and device constraints, for subsequent storage on-board of a complex device under test (DUT) and activation of a re-generative functional test on the complex device under test (DUT).

18. The computer readable medium as claimed in claim 17, wherein said SBE generation tool comprises:

a random instruction test generator (RIT-G) composer arranged to receive the user directives and the instruction information and generate a compact RIT-G code;

a test execution directive composer arranged to receive the user directives and the device constraints and create a run time environment needed to enable the re-generative functional test to repeatedly generate functional tests and execute generated tests on-board the complex device under test (DUT);

a test result compaction module composer arranged to generate a test result compaction module code; and

a code merger arranged to merge code from the RIT-G composer, the test execution directive composer and the test result compaction module composer to generate the software

built-in self-test engine (SBE).

19. The computer readable medium as claimed in claim 18, wherein said SBE is merged with an expected test result and then loaded on-board a complex device under test (DUT) so as to activate a re-generative functional test on the complex device under test (DUT) and make a comparison between test results of the re-generative functional test and the expected test result to check for design validations and/or manufacturing defects.

20. The computer readable medium as claimed in claim 19, wherein said expected test result is obtained from computer modeling of the complex device under test (DUT) or from a known good device.

21. The computer readable medium as claimed in claim 18, wherein said run time environment includes a test execution environment which employs an exception handler for handling illegal conditions such as undesirable memory accesses, deadlock, shut-down, and infinite loops, and a RIT environment which provides equivalent operating system (OS) functions needed by the RIT generator to generate the re-generative functional test.

22. The computer readable medium as claimed in claim 18, wherein said compact RIT-G code produced is a C-language program which is compiled by a C-compiler to produce an

assembly language version of the RIT-G code, and when the run time environment, the test result compaction module code and the assembly language version of the RIT-G code are assembled by an assembler, a single program indicating the SBE in the target DUT's object code is obtained.

23. The computer readable medium as claimed in claim 18, wherein said compact RIT-G code includes an instruction generation module for generating individual instructions during testing application.

24. The computer readable medium as claimed in claim 17, wherein said software built-in self-test engine (SBE) as generated comprises:

a RIT generator configured with compact RIT machine code that can reside on-board the complex device under test (DUT) for generating the re-generated functional test;

a test program execution module configured with test execution directives for providing a run time environment to store and run the re-generated functional test; and

a test result compaction module configured with compression machine code that compresses test results of the re-generated functional test for storage on-board the complex device under test (DUT).

25. The computer readable medium as claimed in claim 17, wherein said software built-in self-test engine (SBE) is programmed to generate and execute one or more ("N")

instruction sequences during testing, each sequence being executed on one or more (M) data sets, where N and M represent an integer no less than "1" and are user-specified numbers used in generating the SBE by the SBE generation tool.

26. The computer readable medium as claimed in claim 25, wherein said software built-in self-test engine (SBE) is further programmed to generate one or more signatures to provide a unique identification of the test result of each test sequence and indicate whether the test result of a particular test sequence is "good" or "bad".

27. A method for generating a software built-in self-test engine (SBE) for on-chip generation and application of a re-generative functional test on a complex device under test (DUT), comprising:

obtaining user directives which indicate user desired actions;

obtaining instruction information which defines a suite of instructions; and

generating a software built-in self-test engine (SBE) based on the user directives, the instruction information and device constraints, for subsequent storage on-board of a complex device under test (DUT) and activation of a re-generative functional test on the complex device under test (DUT).

28. The method as claimed in claim 27, wherein said software built-in self-test engine

(SBE) is generated by:

generating a compact random instruction test generator (RIT-G) code based on the user directives and the instruction information;

creating a run time environment needed to enable the re-generative functional test to repeatedly generate functional tests and execute generated tests on-board the complex device under test (DUT) based on the device constraints;

generating a test result compaction module code based on the user directives and the device constraints; and

merging the RIT-G code, the run time environment and the test result compaction module code to obtain the software built-in self-test engine (SBE).

29. The method as claimed in claim 27, wherein said SBE is merged with an expected test result and then loaded on-board a complex device under test (DUT) so as to activate a re-generative functional test on the complex device under test (DUT) and make a comparison between test results of the re-generative functional test and the expected test result to check for design validations and/or manufacturing defects.

30. The method as claimed in claim 29, wherein said expected test result is obtained from computer modeling of the complex device under test (DUT) or from a known good device.

31. The method as claimed in claim 28, wherein said run time environment includes a test execution environment which employs an exception handler for handling illegal conditions such as undesirable memory accesses, deadlock, shut-down, and infinite loops, and a RIT environment which provides equivalent operating system (OS) functions needed by the RIT generator to generate the re-generative functional test.

32. The method as claimed in claim 28, wherein said compact RIT-G code produced is a C-language program which is compiled by a C-compiler to produce an assembly language version of the RIT-G code, and when the run time environment, the test result compaction module code and the assembly language version of the RIT-G code are assembled by an assembler, a single program indicating the SBE in the target DUT's object code is obtained.

33. The method as claimed in claim 28, wherein said complex device under test (DUT) indicates a microprocessor.

34. The method as claimed in claim 28, wherein, when test patterns of the SBE are applied to the microprocessor from an on-board memory, the microprocessor performs the following:

beginning a set-up for executing test patterns;

executing the test patterns to generate a series of test sequences and associated data for

respective test sequences;

running the test sequences, and at the end of the test sequences, obtaining the test results for storage in the on-board memory; and

dumping the test results of the test patterns for making a comparison with the expected test result to check for design validations and/or manufacturing defects.

35. The method as claimed in claim 34, wherein said software built-in self-test engine (SBE) is programmed to generate and execute one or more ("N") instruction sequences, each sequence being executed on one or more (M) data sets during testing, where N and M represent an integer no less than "1" and are user-specified numbers used in generating the SBE.

36. The method as claimed in claim 35, wherein said software built-in self-test engine (SBE) is further programmed to generate one or more signatures to provide a unique identification of the test result of each test sequence and indicate whether the test result of a particular test sequence is "good" or "bad".